In [73]:

```
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from pylab import *
import scipy
from scipy.stats import *
```

In [71]:

```
# Подбор вторых параметров для распределений
p_1 = np.random.uniform(0.5,1,1)[0]
p_2 = np.random.uniform(0.5,1,1)[0]
p_3 = np.random.uniform(0.5,1,1)[0]
```

Биномиальное распределение

In [118]:

```
# Число элементов выборки - 1
N = 1001
# Первый параметр
m = 50


s_all = binom.rvs(n=m, p=p_1, size=N)

effect = []
for n in range(1, N + 1):
    effect += [s_all[:n].sum() * 1.0 / n / m]

# 1.0 / информация фишера
inf_fish = []
for n in range(1, N):
    inf_fish += [effect[n] * (1.0 - effect[n]) * 1.0 / m / n]
```

In [119]:

```
# Число бутстрепных выборок
K = 500
boot = []
for n in range(1, N):
    if n % 200 == 0:
        print n
    disp = []
    for i in range(1, K + 1):
        s_boo = binom.rvs(n=m, p=effect[:n], size=n)
        disp += [s_boo.sum() * 1.0 / n / m]
    boot += [var(disp)]
```

```
200
400
600
800
1000
```

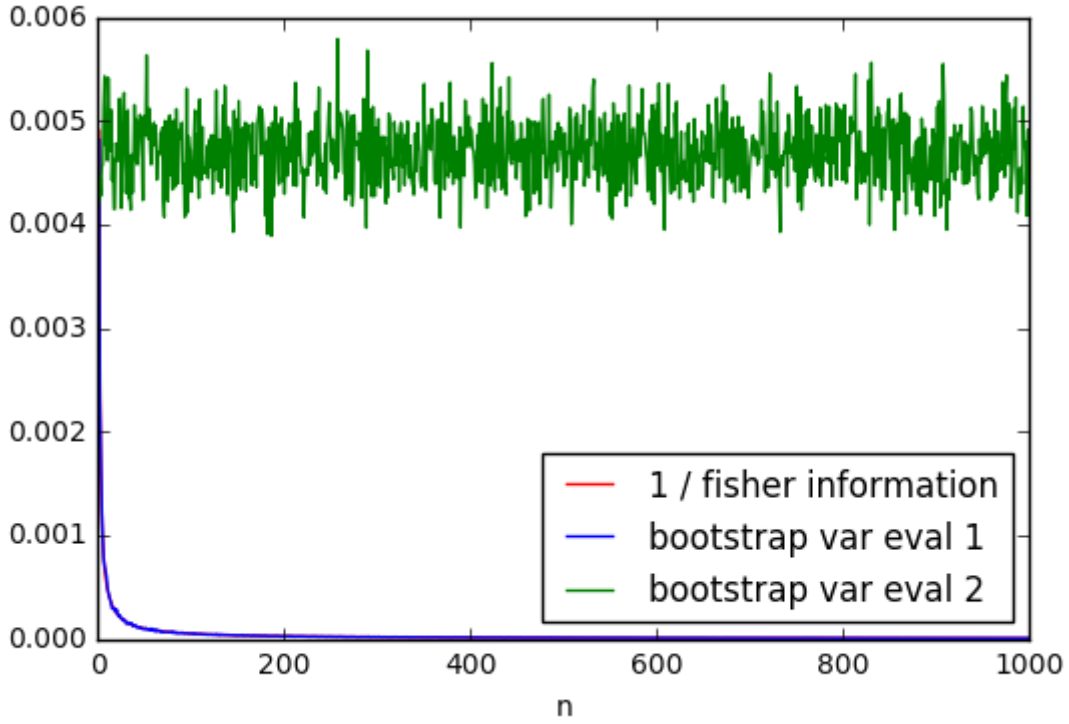
In [120]:

```
# Число бутстрепных выборок
K = 500
boot_1 = []
for n in range(1, N):
    if n % 200 == 0:
        print n
    disp = []
    for i in range(1, K + 1):
        s_boo = binom.rvs(n=m, p=effect[:n], size=n)
        disp += [s_boo[0] * 1.0 / m]
    boot_1 += [var(disp)]
```

```
200
400
600
800
1000
```

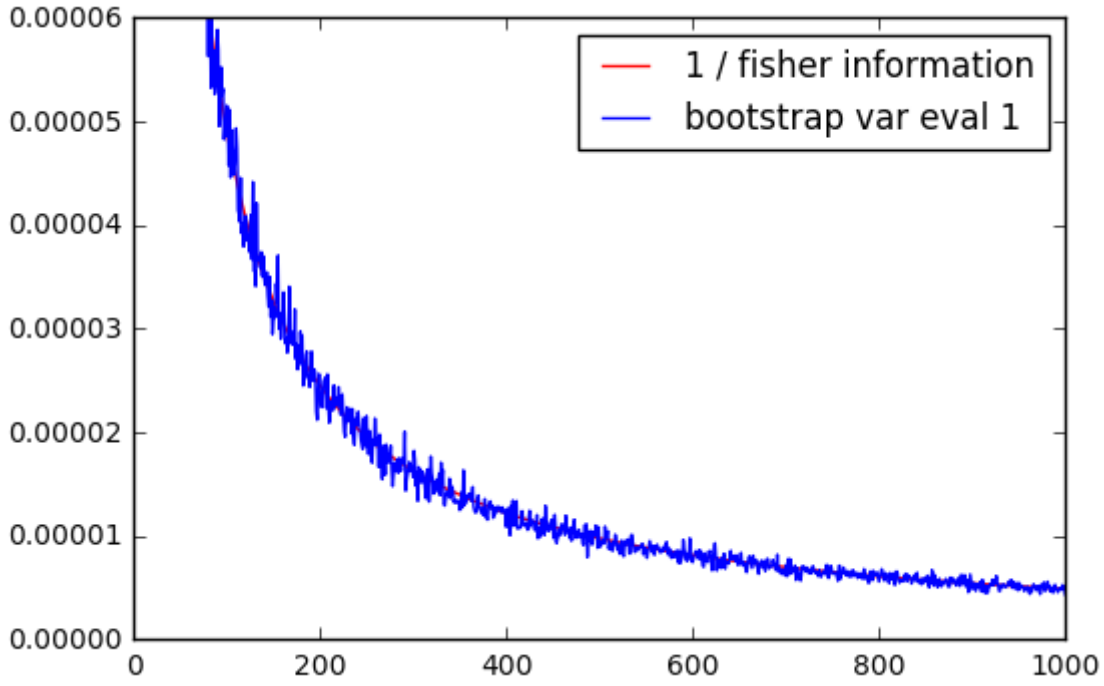
In [138]:

```
ylim(0, 0.006)

plt.plot(range(1, N), inf_fish, color = 'red', label='1 / fisher information')
plt.legend(loc='best')
plt.plot(range(1, N), boot, color = 'blue', label='bootstrap var eval 1')
plt.legend(loc='best')
plt.plot(range(1, N), boot_1, color = 'green', label='bootstrap var eval 2')
plt.legend(loc='best')

plt.xlabel('n')

plt.show()
```



In [144]:

```
ylim(0, 0.00006)

plt.plot(range(1, N), inf_fish, color = 'red', label='1 / fisher information')
plt.legend(loc='best')
plt.plot(range(1, N), boot, color = 'blue', label='bootstrap var eval 1')
plt.legend(loc='best')
plt.show()
```



Экспоненциальное распределение

In [148]:

```
# Число элементов выборки - 1
N = 1001

s_all = expon.rvs(p_2, size=N)

effect = []
for n in range(1, N + 1):
    effect += [s_all[:n].sum() * 1.0 / n]

# 1.0 / информация фишера
inf_fish = []
for n in range(1, N):
    inf_fish += [effect[n] ** 2.0 * 1.0 / n]
```

In [150]:

```
# Число бутстрепных выборок
K = 500
boot = []
for n in range(1, N):
    if n % 200 == 0:
        print n
    disp = []
    for i in range(1, K + 1):
        s_boo = expon.rvs(effect[:n], size=n)
        disp += [s_boo.sum() * 1.0 / n]
    boot += [var(disp)]
```

```
200
400
600
800
1000
```

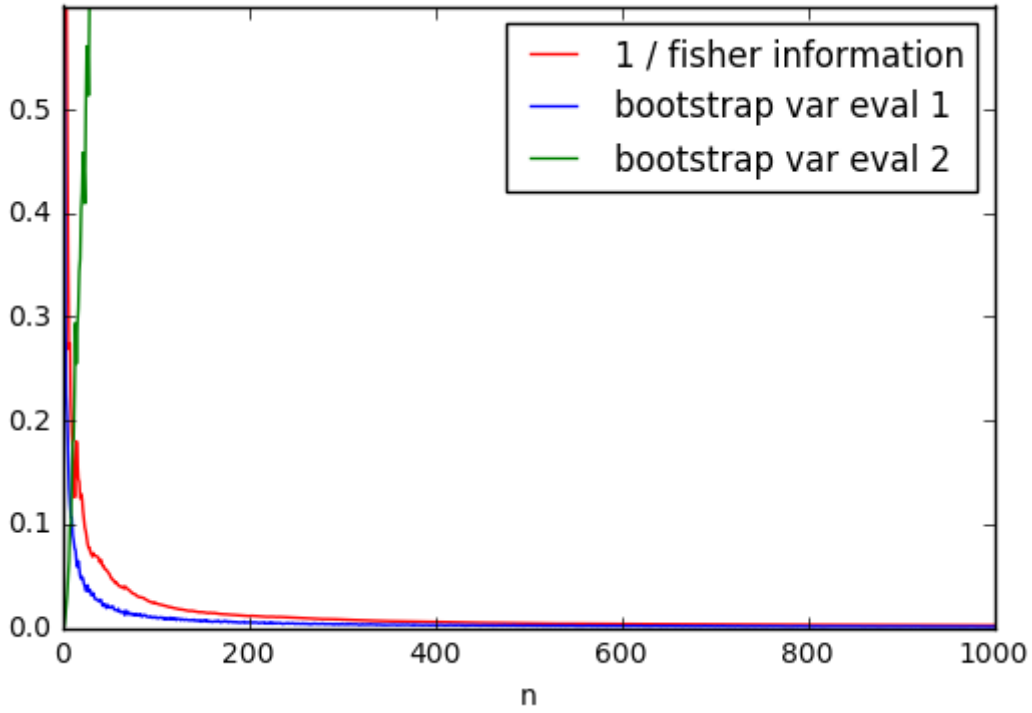
In [156]:

```
# Число бутстрепных выборок
K = 500
boot_1 = []
for n in range(1, N):
    if n % 200 == 0:
        print n
    disp = []
    for i in range(1, K + 1):
        s_boo = expon.rvs(effect[:n], size=n)
        disp += [(n - 1)* 1.0 / (s_boo.sum() * 1.0 / n)]
    boot_1 += [var(disp)]
```

```
200
400
600
800
1000
```

In [158]:

```
ylim(0, 0.6)

plt.plot(range(1, N), inf_fish, color = 'red', label='1 / fisher information')
plt.legend(loc='best')
plt.plot(range(1, N), boot, color = 'blue', label='bootstrap var eval 1')
plt.legend(loc='best')
plt.plot(range(1, N), boot_1, color = 'green', label='bootstrap var eval 2')
plt.legend(loc='best')

plt.xlabel('n')

plt.show()
```



Нормальное распределение

In [160]:

```
# Число элементов выборки - 1
N = 1001
# Первый параметр
sig = 2.1


s_all = norm.rvs(p_3, sig, size=N)

effect = []
for n in range(1, N + 1):
    effect += [s_all[:n].sum() * 1.0 / n]

# 1.0 / информация фишера
inf_fish = []
for n in range(1, N):
    inf_fish += [sig / n]
```

In [161]:

```
# Число бутстрепных выборок
K = 500
boot = []
for n in range(1, N):
    if n % 200 == 0:
        print n
    disp = []
    for i in range(1, K + 1):
        s_boo = norm.rvs(effect[:n], sig, size=n)
        disp += [s_boo.sum() * 1.0 / n]
    boot += [var(disp)]
```

```
200
400
600
800
1000
```

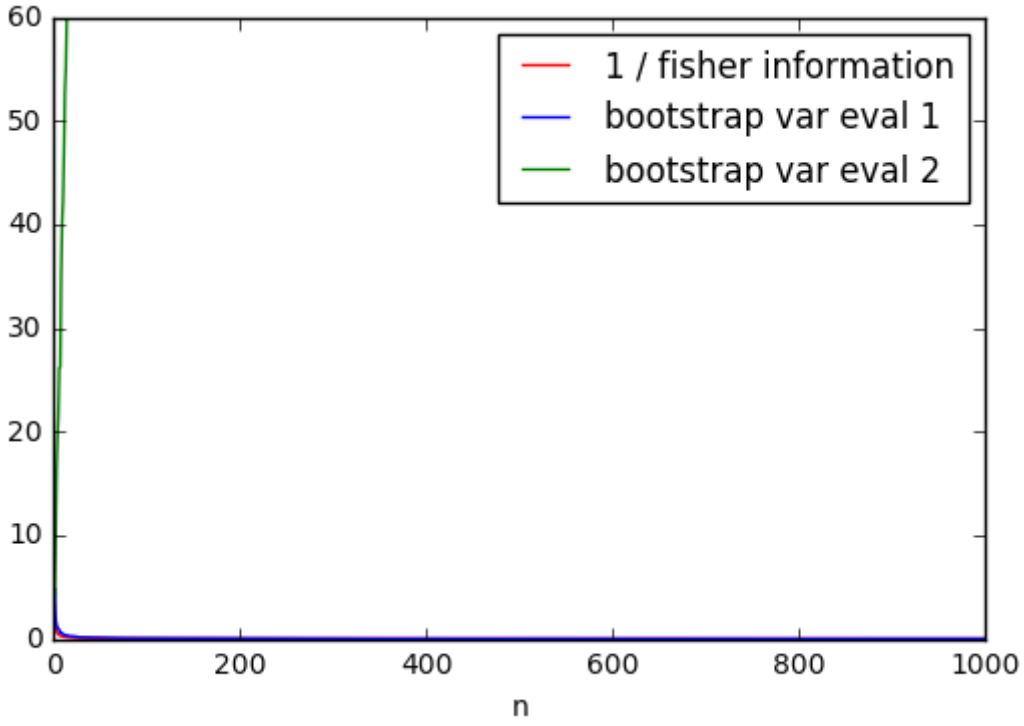
In [162]:

```
# Число бутстрепных выборок
K = 500
boot_1 = []
for n in range(1, N):
    if n % 200 == 0:
        print n
    disp = []
    for i in range(1, K + 1):
        s_boo = norm.rvs(effect[:n], sig, size=n)
        disp += [median(s_boo.sum())]
    boot_1 += [var(disp)]
```

```
200
400
600
800
1000
```

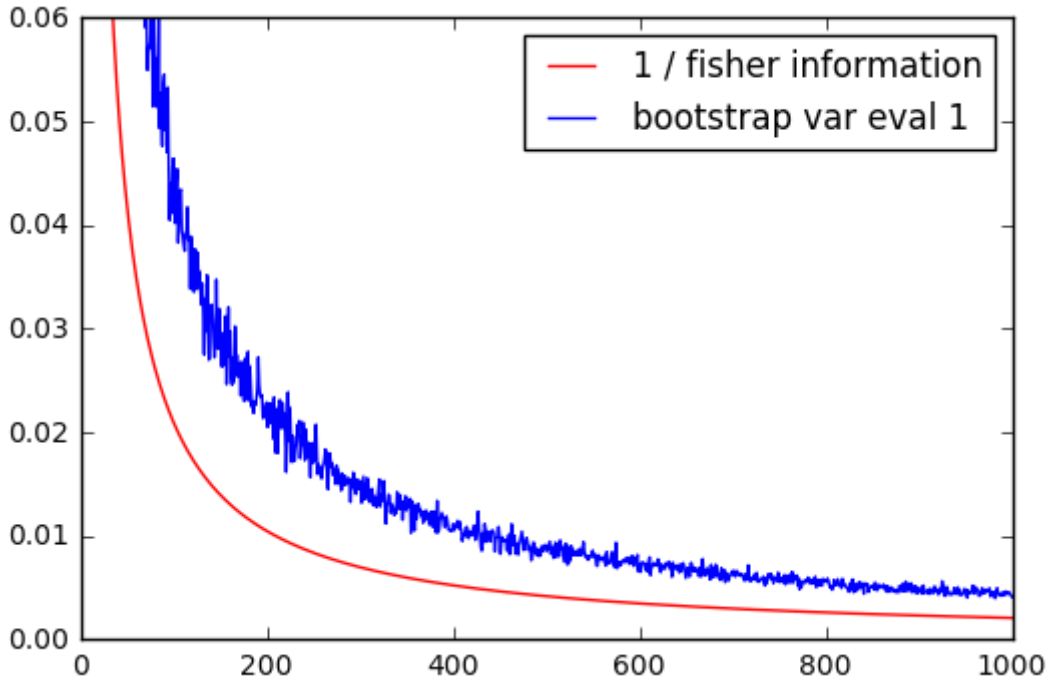
In [166]:

```
ylim(0, 60)

plt.plot(range(1, N), inf_fish, color = 'red', label='1 / fisher information')
plt.legend(loc='best')
plt.plot(range(1, N), boot, color = 'blue', label='bootstrap var eval 1')
plt.legend(loc='best')
plt.plot(range(1, N), boot_1, color = 'green', label='bootstrap var eval 2')
plt.legend(loc='best')

plt.xlabel('n')

plt.show()
```



In [168]:

```
ylim(0, 0.06)

plt.plot(range(1, N), inf_fish, color = 'red', label='1 / fisher information')
plt.legend(loc='best')
plt.plot(range(1, N), boot, color = 'blue', label='bootstrap var eval 1')
plt.legend(loc='best')
plt.show()
```



Вывод

Эффективные оценки получились лучше, чем просто несмещенные. Неравенство Рао-Крамера выполнено